CS2102 Analysis and Design of Algorithms **24 de abril de 2019**

Quicksort

Bryan Gonzales Vega University of Engineering and Technology

I. Instructions

Complete the following questions and attach your source code, LaTeX, and other tools (Dockerfile, images, etc) you required to complete the problems using your GitHub Classroom repository.

II. Analysis

We already prove that Quicksort algorithm runs at $\Theta(nlogn)$ time for the average and best case. Nevertheless for the worst case it runs at $\Theta(n^2)$. Although we have a solid understanding of the common implementation there are still some improvements we can do to it. The following exercises give you some hints about how you can improve the Quicksort algorithm. For all the exercises run some simulations to test out your answers.

1. Can you find the minimum number of elements n for which the Quicksort algorithm always runs faster? If so implement it and simulate some scenarios where you can see the improvement.

2. Besides time a recursive algorithm consumes memory. Explain the memory consumption of a recursive algorithm and give some examples explaining how it works.

3. What is tail recursion? Is it faster than the regular recursion ? Try it out with the Quicksort algorithm. Does it run faster?

4. How can the Quicksort algorithm be implemented in a parallel way ? Propose your pseudo-code for that task and sketch the execution time.

5. Propose one improvement to the Quicksort algorithm in terms of space or execution time.

6. Quicksort requires a pivot element that could be chosen in different ways like: i) always the first element, ii) the last element, iii) the middle one or iv) a random element. Is there any difference in terms of performance ? If so specify for which cases.

III. Inversions

In the resources folder you will find a text file with 100,000 integers between 1 and 100,000 (inclusive) in some order, with no integer repeated. Your task now is to compute the number of

inversions of the given file. Compare the execution times of the naive and divide and conquer implementations. As an extra you can compare both implementations with some other strategy. Write the number of inversions and plot your results.

IV. Bonus

Since we are still on divide and conquer strategy, use it to solve the 10810-UVa problem and paste your code and result in here.